REMARKS

Introduction

Claims 1 and 20 have been amended. New claims 39-50 have been added. Claims 14-19 and 33-38 were previously cancelled. The application now includes claims 1-13, 20-32, and 39-50. In view of the RCE filed herewith in compliance with 37 CFR 1.114 and 1.198, reconsideration of the rejection of the application is respectfully requested in view of the claim amendments and the following remarks.

Applicants thank the Examiner, and his supervisor, for taking the time to conduct a telephonic interview with the Applicants' representative on March 21, 2011. The substance of the interview is reflected in this Amendment.

The Claims are Allowable because the Prior Art Fails to Disclose Invoking a Second Code Statement Processing Unit of a Second Programming Language to Process a Code Statement, when a First Code Statement Processing Unit Locates a Code Statement of the Second Programming Language within a First Code Section and Invokes an Execution Engine recursively

Claims 1-3, 6-7, 20-22, and 25-26 were rejected under 35 U.S.C. § 102(e) as allegedly being anticipated by Wang, U.S. Patent No. 6,292,936 ("Wang"). Claims 4-5, 8, 23-24, and 27 were rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over Wang, in view of Claussen et al., U.S. Patent No. 6,732,330 ("Claussen"). Claims 9-13 and 28-32 were rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over Wang, in view of Conner et al., U.S. Patent No. 5,428,792 ("Conner"). Reconsideration of these rejections is respectfully requested because the prior art fails to disclose invoking a second code statement processing unit

of a second programming language to process a code statement, when a first code statement processing unit locates a code statement of the second programming language within a first code section and invokes a execution engine recursively.

In one embodiment, a computing environment is provided with an execution engine, supplemented with a number of language specific processing units to facilitate execution of data processing representations expressed with programming instructions of multiple programming languages. *See e.g.*, U.S. Patent Publication No. 2004/0040011 (*i.e.*, "Specification") at paragraph [0022]; Fig. 1. According to the embodiment, upon encountering a code section/statement of a language, the execution engine invokes the corresponding language specific processing unit to augment and provide the language specific processing required to process and facilitate execution of the code section/statement. *See e.g.*, Specification at paragraph [0025]; Fig. 1. Upon encountering a sub-section written in an unknown programming language, the language specific processing unit can delegate the processing of the sub-section back to the execution engine. *See e.g.*, Specification at paragraph [0026]; Fig. 1. The execution engine, in turn can pass the sub-section to an appropriate language specific processor and return the result to the requesting language specific processing unit. *See e.g.*, Specification at *id.*

Wang describes a distributed computer system that includes server system 106 executing Web daemons 108. *See* Wang at col. 2, lines 36-48; FIG 1. Each web daemon 108 includes runtime processors 110 and 112 (i.e., Java Virtual Machine (JVM) 110 that executes Java programming statements and a VisualBasic Script interpreter

(VBSI) 112 that executes VisualBasic Script programming statements). *See* Wang at col. 2, lines 49-55; FIG. 1. The server system also includes one or more translators 114, where each translator 114 (e.g., HTML Parser 114) interprets an original input source 116 comprising scripts programming instructions, etc., and translates the original input source 116 into two intermediate sources (i.e., intermediate source 200 and intermediate source 202). *See* Wang at col. 2, lines 56-67; col. 3, lines 25-30; FIGS. 1 and 2. To create the intermediate source 200, the HTML Parser 114 translates a first VisualBasic Script block in the original input source 116 into a thread object run method, a notify method, and an immediate wait method. *See* Wang at col. 3, lines 31-40; FIG. 2. The remaining VisualBasic Script blocks in the original input source 116 are translated into notify method and wait method invocations. *See* Wang at col. 3, lines 40-43; FIG. 2. To create the intermediate source 202, the HTML Parser 114 translates every HTML block after the first Visual Basic Script block into a synchronizer token. *See* Wang at col. 3, lines 44-46. During runtime, the JVM 110 and the VBSI 112 execute the respective intermediate sources 200 and 202. *See* Wang at col. 3, lines 50-52.

Wang discloses that the original input source 116, which contains HTML text (including Java source code) with embedded VisualBasic Script statements, is translated into intermediate sources 200 and 202, and thus neither JVM 110 (which the Office Action interprets as a "first code statement processing unit") nor VSBI 112 (which the Office Action interprets as a "second code statement processing unit") executes original input source 116. Instead, JVM 110 executes intermediate source 200 which

only contains HTML text (including Java source code) and thread object methods, and VBSI 112 executes intermediate source 202 which only contains VisualBasic Script statements and synchronization objects. Thus, even if JVM 110 and VSBI 112 could be considered analogous to a "first code statement processing unit" and a "second code statement processing unit," respectively (not admitted by Applicants), JVM 110 does not invoke VSBI 112 upon locating a block of a VisualBasic Script programming language within intermediate source 200. Instead, JVM 110 invokes VSBI 112 when it finds a thread object method. Likewise, VSBI 112 does not invoke JVM 110 upon locating a block of HTML (including Java source code). Instead, VSBI 112 invokes JVM 110 when it finds a synchronization token. Thus, Wang fails to disclose or suggest, locating a code statement of a second programming language within a first code section.

As Wang fails to disclose or suggest this limitation, Wang also fails to disclose or suggest invoking an execution engine recursively upon locating a code statement of a second programming language within a first code section. Furthermore, Wang also fails to disclose or suggest invoking a second code statement processing unit of a second programming language to process a code statement, upon locating a code statement of the second programming language within a first code section and invoking a execution engine recursively.

Furthermore, neither Claussen nor Conner cure the deficiencies of Wang. Claussen describes a page handling framework where different scripting languages may reside side-by-side, or nested within each other on the same web page. *See* Claussen at col. 2, lines 58-60. Claussen fails to disclose or suggest invoking a second

code statement processing unit of a second programming language to process a code statement, when a first code statement processing unit locates a code statement of the second programming language within a first code section and invokes a execution engine recursively. Conner describes a system for defining language dependent object definitions as a neutral set of information from which object support for any language, including support between languages, is provided. *See* Conner at Abstract. Similar to Claussen, Conner fails to disclose or suggest invoking a second code statement processing unit of a second programming language to process a code statement, when a first code statement processing unit locates a code statement of the second programming language within a first code section and invokes a execution engine recursively.

In contrast to the cited prior art, independent claim 1 recites "invoking, by the execution engine, a second code statement processing unit of a second programming language to process a code statement, when the first code statement processing unit locates a code statement of the second programming language within the first code section, and invokes the execution engine recursively," and "invoking, by the execution engine, the first code statement processing unit of the first programming language to process a code statement, when the second code statement processing unit locates a code statement of the first programming language within the second code section, and invokes the execution engine recursively." For at least these reasons amended independent claim 1, amended independent claim 20 (which recites similar limitations) and new independent claim 39 (which also recites similar limitations), should now be

allowable over the cited prior art. The remaining claims depend from one of the above independent claims and should also be allowable for at least the above reasons.

Conclusion

Applicants respectfully request favorable action in connection with this application.

The Examiner is invited and urged to contact the undersigned to discuss any matter concerning this application.

An RCE fee is required for this submission. Should any other fee be required, the Commissioner is authorized to charge any such fee to Counsel's Deposit Account 50-2222.

Respectfully submitted,

Date: March 23, 2011

/Keith M. Mullervy/
Keith M. Mullervy
Attorney for Applicants
Registration No. 62,382

Customer No. 74739
SQUIRE, SANDERS & DEMPSEY (US) LLP
14TH Floor
8000 Towers Crescent Drive
Vienna, Virginia 22182-6212
Telephone: 703-720-7876
Fax: 703-720-7802

KMM:mmi